Clustering community networks by Variable neighborhood search

Nenad Mladenović,

Mathematical Institute, SANU, Belgrade

LAMIH, University of Valenciennes, France

(joint work with J Brimberg, S Cafieri, E Carrizosa, P Hansen, R Todosijević and D Urošević)

- Variable neighborhood search (VNS) algorithms
- Ratio-cut clustering on the network with VNS (Cafieri, Hansen, Ml)
- Multi diversity grouping with VNS (Brimberg, Urošević, Ml)
- Clique partitioning with VNS (Brimberg, Urošević, Ml)
- Sum-of-squared-distances on networks and clustering with VNS (Carrizosa, Todosijević, Ml)

Introduction

- Mladenovic (1995) - Variable neighborhood algorithm - a new metaheuristic for combinatorial optimization.
- Mladenovic, Hansen (1997) Variable neighborhood search.
- This paper cited around 2,000 times (Google scholar).
- This paper cited > 700 times (Web of Knowledge)
- Euro Mini Conference devoted to VNS (2005);
- Euro Mini Conference on VNS (2012), 80 contributions.
- 6 VNS special issues (JOGO, COR, JOH, EJOR, YUJOR and IMA-MAN).
- 3rd International conference on VNS, Djerba, Tunis 2014. (www.vns-metaheuristic.com)

- Several book Chapters.

Variable neighborhood search

- Let $\mathcal{N}_k$, $(k = 1, \ldots, k_{max})$,
  a finite set of pre-selected neighborhood structures,
- $\mathcal{N}_k(x)$ the set of solutions in the k^{th}
  neighborhood of x.
- An optimal solution x_{opt} (or global minimum) is a
  feasible solution where a minimum is reached.
- We call $x' \in X$ a local minimum with respect to
  $\mathcal{N}_k$ (w.r.t. $\mathcal{N}_k$ for short), if there is no
  solution $x \in \mathcal{N}_k(x') \subseteq X$ such that $f(x) < f(x')$.
- VNS is based on three simple facts:
  - ▷ A local minimum w.r.t. one neighborhood structure is not necessarily so for another;
  - ▷ A global minimum is a local minimum w.r.t. all possible neighborhood structures;
  - ▷ For many problems, local minima w.r.t. one or several $\mathcal{N}_k$ are relatively close to each other.

Variable neighborhood search

- In order to solve optimization problem by using several neighborhoods, facts 1 to 3 can be used in three different ways:
  - (i) deterministic;
  - (ii) stochastic;
  - (iii) both deterministic and stochastic.

- Some VNS variants
  - Variable neighborhood descent (VND) (sequential, nested)
  - Reduced VNS (RVNS)
  - Basic VNS (BVNS)
  - Skewed VNS (SVNS)
  - General VNS (GVNS)
  - VN Decomposition Search (VNDS)
  - Parallel VNS (PVNS)
  - Primal Dual VNS (P-D VNS)
  - Exterior point VNS;
  - VN Branching
  - VN Pump and VN Diving;
  - Continuous VNS
  - Mixed Nonlinear VNS (RECIPE), etc.

Ratio-cut divisive clustering - Introduction

- Complex systems in a variety of domains are represented by networks: social, telecommunication, transportation, biological networks, and many more.
- A topic of particular interest is the identification of clusters or communities.
- Several models and clustering criteria have been proposed. The most used is *modularity*,
- Let $S \subseteq V$ be a subset of vertices. Then the degree k_i can be separated into two components $k_i^{in}(S)$ and $k_i^{out}(S)$.
- A set of vertices S forms a community in the strong sense iff every one of its vertices has more neighbors within the community than outside:

$$k_i^{in}(S) > k_i^{out}(S), \quad \forall i \in S.$$

- A set of vertices S forms a community in the weak sense iff the sum of all degrees within S is larger than the sum of all degrees joining S to the rest of the network:

$$\sum_{i \in S} k_i^{in}(S) > \sum_{i \in S} k_i^{out}(S).$$

Edge-ratio criterion

- The ratio of the number of edges within a community to the number of cut edges which have one end point only within that community is considered:

$$r(S) = \sum_{i \in S} k_i^{in}(S) / \sum_{i \in S} k_i^{out}(S).$$

- When dividing S, this ratio for both communities S_1 and S_2 is considered and the smallest value is maximized:

$$\max_{S_1, S_2 \subset V} \min\left(r(S_1), r(S_2)\right).$$

  where $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$, $S_1, S_2 \neq \emptyset$.

- $$f(S_1, S_2) = \max_{S_1, S_2 \in \mathcal{P}} \min\{f_1(S_1), f_2(S_2)\} \quad (1)$$

  where $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$.

- This problem is NP hard (Noble et al 2011).

VNS for divisive hierarchical Ratio-cut clustering

- Solution is represented in the following way:
  - $x = \{x_1, \ldots, x_{|S|}\}$ – indices of vertices that belong to S.
  - $\delta_{x_j} = \begin{cases} 1 & \text{if} \quad x_j \in S_1 \\ 0 & \text{if} \quad x_j \in S_2 \end{cases}$

- To efficiently update objective function values we need:
  - n_1 – the number of vertices in set S_1;
  - n_2 – the number of vertices in set S_2;
  - n_3 – the number of vertices in set S_3;
  - m_1 – the number of edges in subgraph G_1;
  - m_2 – the number of edges in subgraph G_2;
  - c_1 – the number of cut edges with one end point in S_1 and another in $V \setminus S_1 = S_2 \cup S_3$;
  - c_2 – the number of cut edges with one end point in S_2 and another in $V \setminus S_2 = S_1 \cup S_3$;
  - k_i^{in} – inner degree of vertex v_i (both end points belong to either S_1 or S_2);
  - k_i^{cut} – cut (outer) degree of vertex v_i (the number of links of v_i with vertices from the different sets);
  - $f_1 = 2 \cdot \frac{m_1}{c_1}$ – ratio for subgraph G_1;
  - $f_2 = 2 \cdot \frac{m_2}{c_2}$ – ratio for subgraph G_2;
  - $f = \min\{f_1, f_2\}$ – objective function value.

- Note that all values from above are known if partition (S_1, S_2, S_3) is known $(S_j \in \mathcal{P})$. Therefore, to be more precise, instead of n_1 we can put $n_1(S_1)$, instead of m_2 we can put $m_2(S_2)$, etc.

- At each hierarchy level ℓ, we calculate the "rest" degree of each vertex from S^ℓ as

$$k_i^{rest} = k_v - k_v(S), v \in V.$$

- The degree of each vertex can be presented as

$$k_v = k_v^{in} + k_v^{cut} + k_v^{rest},$$

Allocation local search

- The basic local change we use is allocation move: take some entity v from $S = S_1 \cup S_2$ and change its allocation: if $v \in S_1$ then, after the allocation move, it will belong to S_2 and vice versa.

- In the Algorithm `Move-Check`(v, f, δ) input values are the index of entity v that will change its membership, current objective function value f and the current solution δ. We also assume that, together with the solution, values of m_1, m_2, c_1 and c_2 are known as well. The next property gives the new objective function value.

Proposition 1. *Let v be a vertex that moves from one subset to another. If $v \in S_1$ then the new objective function value is*

$$f^{new} = \min\{\frac{2 \cdot (m_1 - k_v^{in})}{c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}}, \frac{2 \cdot (m_2 + k_v^{cut})}{c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}}\}.$$

If $v \in S_2$ then

$$f^{new} = \min\{\frac{2 \cdot (m_1 + k_v^{cut})}{c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}}, \frac{2 \cdot (m_2 - k_v^{in})}{c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}}\}.$$

Proof. A new value of the objective function, by definition is

$$f^{new} = \min\{f_1^{new}, f_2^{new}\} = \min\{\frac{2m_1^{new}}{c_1^{new}}, \frac{2m_2^{new}}{c_2^{new}}\}. \tag{2}$$

If $v \in S_1$ changes its membership to S_2, then the following changes are evident:

$$m_1^{new} = m_1 - k_v^{in} \tag{3}$$

$$m_2^{new} = m_2 + k_v^{cut} \tag{4}$$

Indeed, the new number of inner edges m_1 in S_1 is reduced by k_v^{in} (the inner degree of v) (see (3)); the number of edges in S_2 is augmented by the cut degree of v (see (4)). The numbers of cut edges c_1 and c_2 of S_1 and S_2 need also to be updated:

$$c_1^{new} = c_1 + k_v^{in} - k_v^{cut} - k_v^{rest} \tag{5}$$

$$c_2^{new} = c_2 + k_v^{in} - k_v^{cut} + k_v^{rest} \tag{6}$$

We explain now how (5) is derived. Since v moves to S_2, the number of cut edges c_1^{new} of S_1 is increased by its inner degree in S_1 ($+k_v^{in}$). Also, it should be reduced by k_v^{cut}, since its cut degree contributes to inner degree of S_2. In addition, all vertices from S_3 connected with v after the move to S_2 do not produce cut edges of S_1 any more ($-k_v^{rest}$). Observe that $c_1^{new} = c_2^{new}$ only if $S = V$, i.e., at the first hierarchy level $\ell = 1$ subset $S_3 = \emptyset$ and therefore the $rest$ degrees are equal to 0. Explanation of formula (6) is similar.

If vertex v belongs to S_2 and moves to S_1, then new numbers of inner and cut edges in S_1 and S_2 are

$$m_1^{new} = m_1 + k_v^{cut} \tag{7}$$

$$m_2^{new} = m_2 - k_v^{in} \tag{8}$$

$$c_1^{new} = c_1 + k_v^{in} - k_v^{cut} + k_v^{rest} \tag{9}$$

$$c_2^{new} = c_2 + k_v^{in} - k_v^{cut} - k_v^{rest} \tag{10}$$

Substituting (5) - (10) into equations (2) we get the result. QED.

Pseudo-code

of move evaluation:

Algorithm 1 Allocation move

Function `Move-Check`(v, f, δ)

if δ_v **then**

$mm_1 \leftarrow m_1 - k_v^{in}$; $mm_2 \leftarrow m_2 + k_v^{cut}$
$cc_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}$
$cc_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}$

else

$mm_1 \leftarrow m_1 + k_v^{cut}$; $mm_2 \leftarrow m_2 - k_v^{in}$
$cc_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}$
$cc_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}$

$f_1 \leftarrow \frac{2 \cdot mm_1}{cc_1}$; $f_2 \leftarrow \frac{2 \cdot mm_2}{cc_2}$; $f \leftarrow \min\{f_1, f_2\}$

We denote temporally values of m_1, m_2, c_1 and c_2 with mm_1, mm_2, cc_1 and cc_2, respectively, since we just calculate the new value f in the neighborhood; those values will be changed once we decide to make a move.

Update function

Proposition 2. *Let v be a vertex that change its membership and let r be any adjacent vertex to v ($(v,r) \in E$). Then values of new inner and cut degrees of r are:*

(i) $\bar{k}_r^{in} = k_r^{in} - 1, \quad \bar{k}_r^{cut} = k_r^{cut} + 1$, *if both v and r belong to the same subset;*

(ii) $\bar{k}_r^{in} = k_r^{in} + 1, \quad \bar{k}_r^{cut} = k_r^{cut} - 1$, *if v and r are in different subsets.*

Proof. As mentioned earlier, $r_j = h_{vj}$, $j = 1, \ldots, k_v$ indicate all adjacent vertices of v. There are k_v of them since k_v is the degree of v. Let us denote with $\bar{k}_r^{in}$ and $\bar{k}_r^{cut}$ a new values of inner and cut degrees (after the vertex v is moved from one to another subset of vertices), respectively. Then there are four different cases.

Case 1. $v \in S_1$ and $r_j \in S_1$. Then $\bar{k}_{r_j}^{in} = k_{r_j}^{in} - 1$ and $\bar{k}_{r_j}^{cut} = k_{r_j}^{cut} + 1$;
Case 2. $v \in S_1$ and $r_j \in S_2$. Then $\bar{k}_{r_j}^{in} = k_{r_j}^{in} + 1$ and $\bar{k}_{r_j}^{cut} = k_{r_j}^{cut} - 1$;
Case 3. $v \in S_2$ and $r_j \in S_1$. Then $\bar{k}_{r_j}^{in} = k_{r_j}^{in} + 1$ and $\bar{k}_{r_j}^{cut} = k_{r_j}^{cut} - 1$;
Case 4. $v \in S_2$ and $r_j \in S_2$. Then $\bar{k}_{r_j}^{in} = k_{r_j}^{in} - 1$ and $\bar{k}_{r_j}^{cut} = k_{r_j}^{cut} + 1$.

From above result it is clear that cases 1 and 4 give the same outcome in updating degrees of adjacent vertices to v. The same holds for cases 2 and 3. Thus, we get the result. QED.

Function `Update`(v, f, δ)

if δ_v **then**

$n_1 \leftarrow n_1 - 1;\ n_2 \leftarrow n_2 + 1$
$m_1 \leftarrow m_1 - k_v^{in};\ m_2 \leftarrow m_2 + k_v^{cut}$
$c_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} - k_v^{rest}$
$c_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} + k_v^{rest}$

else

$n_2 \leftarrow n_2 - 1;\ n_1 \leftarrow n_1 + 1$
$m_1 \leftarrow m_1 + k_v^{cut};\ m_2 \leftarrow m_2 - k_v^{in}$
$c_1 \leftarrow c_1 + k_v^{in} - k_v^{cut} + k_v^{rest}$
$c_2 \leftarrow c_2 + k_v^{in} - k_v^{cut} - k_v^{rest}$

$y \leftarrow k_v^{cut};\ k_v^{cut} \leftarrow k_v^{in};\ k_v^{in} \leftarrow y$
for $j = 1, k_v$ **do**

$r \leftarrow h_{ij}$
if $(\delta_r \wedge \delta_v) \vee (\neg\delta_r \wedge \neg\delta_v)$ **then**

$k_r^{cut} \leftarrow k_r^{cut} + 1;\ k_r^{in} \leftarrow k_r^{in} - 1$

else

The question is why we needed Algorithm 1 sice a new solution is obtained in only one command: $\delta_v \leftarrow \neg\delta_v$? The answer is given by the next proposition.

Proposition 3. *The worst case complexity of algorithm* `Update` *is in* $O(k_{max})$*, where* k_{max} *is the maximum degree of* G*.*

Algorithm 2 Best improvement Allocation (re-assignment) local search

```
Function Best-Impr-LS(x, f_opt, δ)
  f_opt ← ∞; improve ← true
  while Improve do
    Improve ← false
    for i = 1, n do
      v ← x_i
      if k_v^cut > 0 then
        Move(v, f, δ)
        if f < f_opt then
          f_opt ← f; w ← v
          improve ← true
    if ¬improve return
    Update(w, f, δ)
```

Proposition 4. *The time complexity of one iteration of* `Best-Impr-LS` *algorithm is* $O(n)$.

Shaking

Algorithm 3 Steps of Shaking operator

Function $\texttt{Shake}(k, \delta)$

$\ell \leftarrow 1$

while $\ell < k$ **do**

$r \leftarrow 1 + (|S| - \ell) \cdot \texttt{Rand}$

$v \leftarrow x_r$

$\texttt{Update}(v, f, \delta)$

$\ell \leftarrow \ell + 1;$

Algorithm 4 Steps of the basic VNS

Function `BVNS` $(A, n, m, \delta, d^{rest}, k_{max}, t_{max})$

repeat

  $k \leftarrow 1$

  repeat

    $\delta' \leftarrow$ `Shake`(k, δ) `/* Shaking */`

    $\delta'' \leftarrow$ `Best-Impr-LS`(δ', f) `/* Local search */`

    $k \leftarrow k + 1$ `/* Next neighborhood */`

    `if` $f(\delta'') < f(\delta)$ `then`

      $\delta \leftarrow \delta''$; $k \leftarrow 1$ `/* Make a move */`

  until $k = k_{max}$

  $t \leftarrow$ `CpuTime()`

until $t > t_{max}$

Function VNS-DHC $(m, n, A, P, k_{max}, t_{max})$

$\ell = 1, S^{\ell} = \{v_1, \ldots, v_n\}$,

$P = \emptyset$, $d^{rest} \leftarrow 0$

Init(m, n, A, H, d)

while $|S^{\ell}| > 1$ **do**

$S_1, S_2 \leftarrow$ BVNS $(S^{\ell}, d^{rest}, k_{max}, t_{max})$ //Find bipartition of S^{ℓ}

if $f \geq 1$ **then**

$P = P \cup \arg\min f(S_1, S_2)$

$S^{\ell} \leftarrow S_1$; $S^{\ell+1} \leftarrow S_2$

$\ell \leftarrow \ell + 1$

Among $\{S^1, \ldots, S^{\ell}\}$, select cluster with the largest cardinality

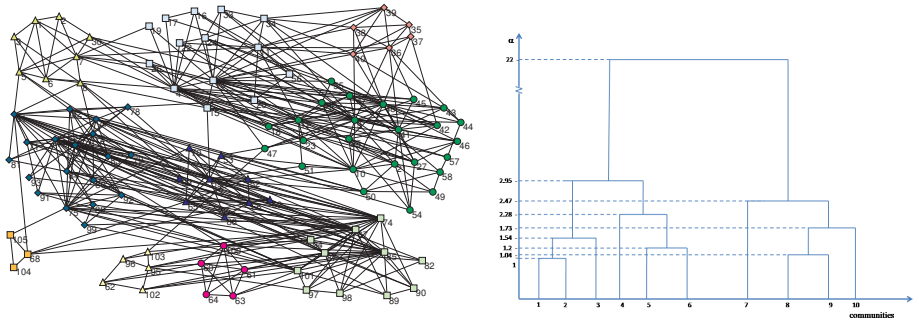
Exchange its position with S^{ℓ}

$d^{rest} = d(G) - d(S^{\ell})$

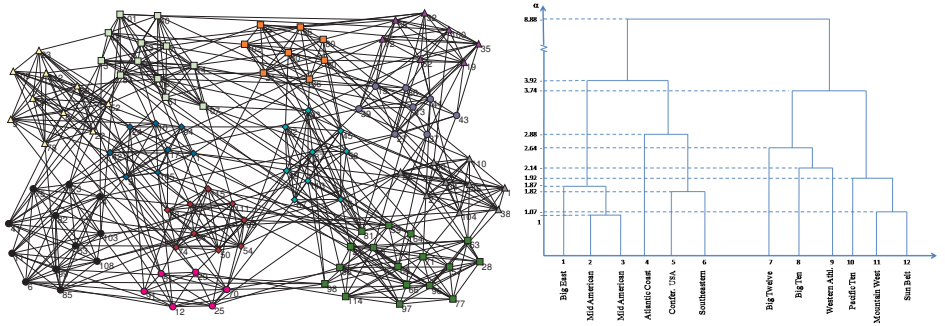
Computational results and analysis

dataset	n	m	Time VNS (sec.)	Time exact (sec.)
karate	34	78	0.0	62.10
dolphins	62	159	9.99e-3	172.2
les miserables	77	254	1.99e-2	283.49
political books	105	441	9.99e-3	716.45
football	115	613	4.99e-2	11780.79
Usair97	332	2126	1.5	3752906.94
netscience_main	379	914	1.17	–
s838	512	819	1.09	–
email	1133	5452	24.75	–
power	4941	6594	40.98	–

Table 1: Comparison of results obtained using VNS and an exact algorithm for bipartition, on datasets from the literature

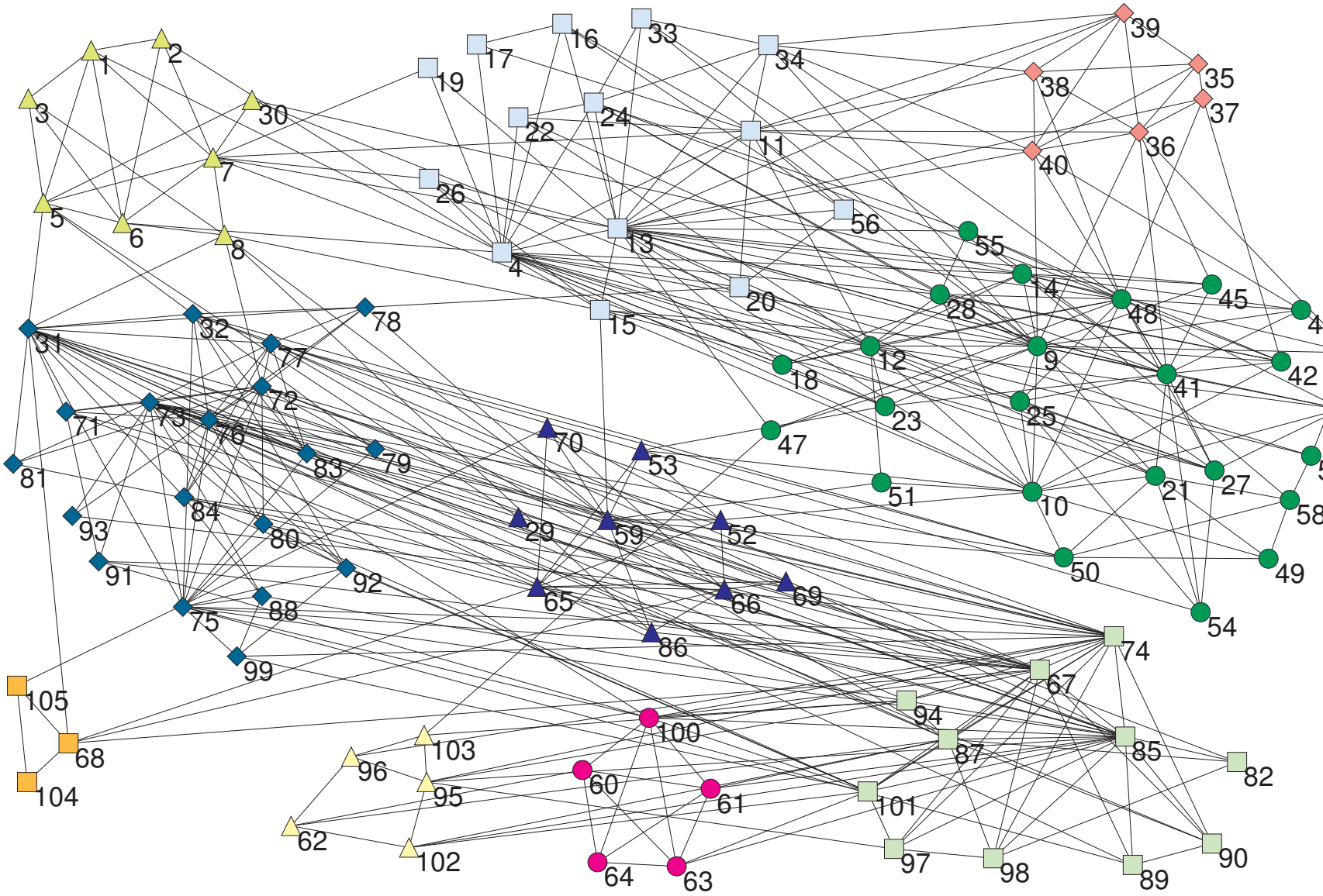


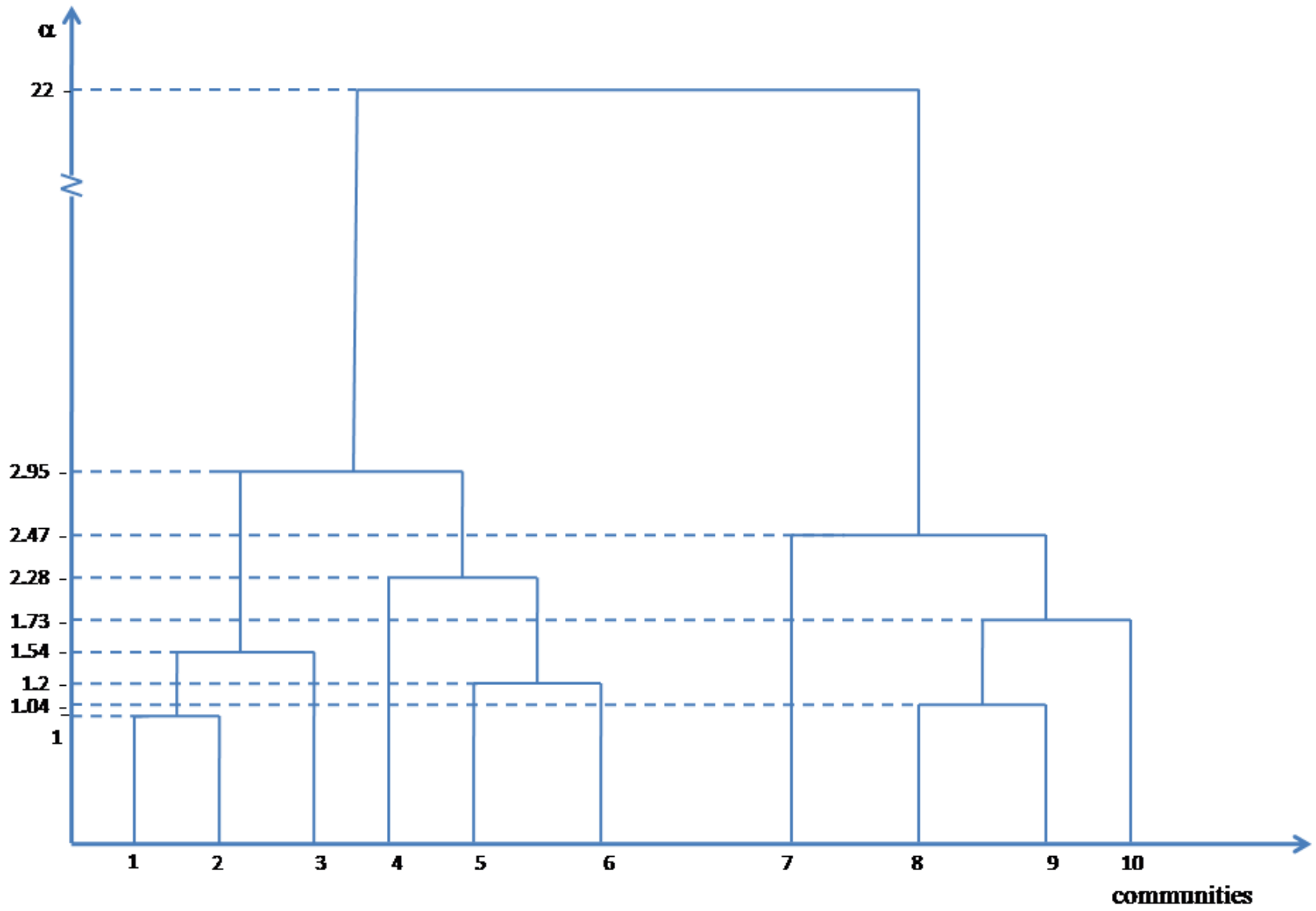
Partition and dendrogram obtained for dataset `polbooks`



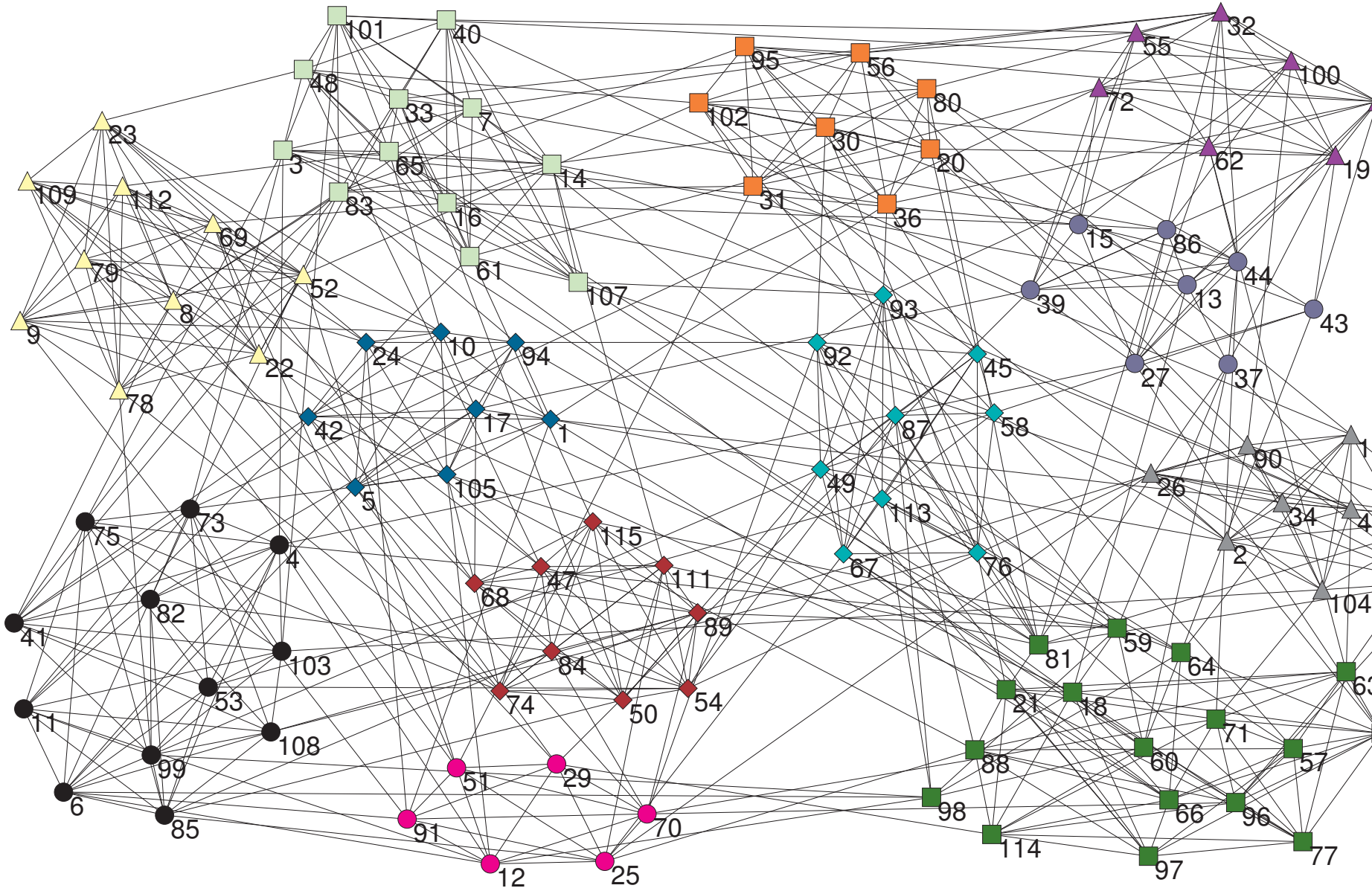
Partition and dendrogram obtained for dataset `football`

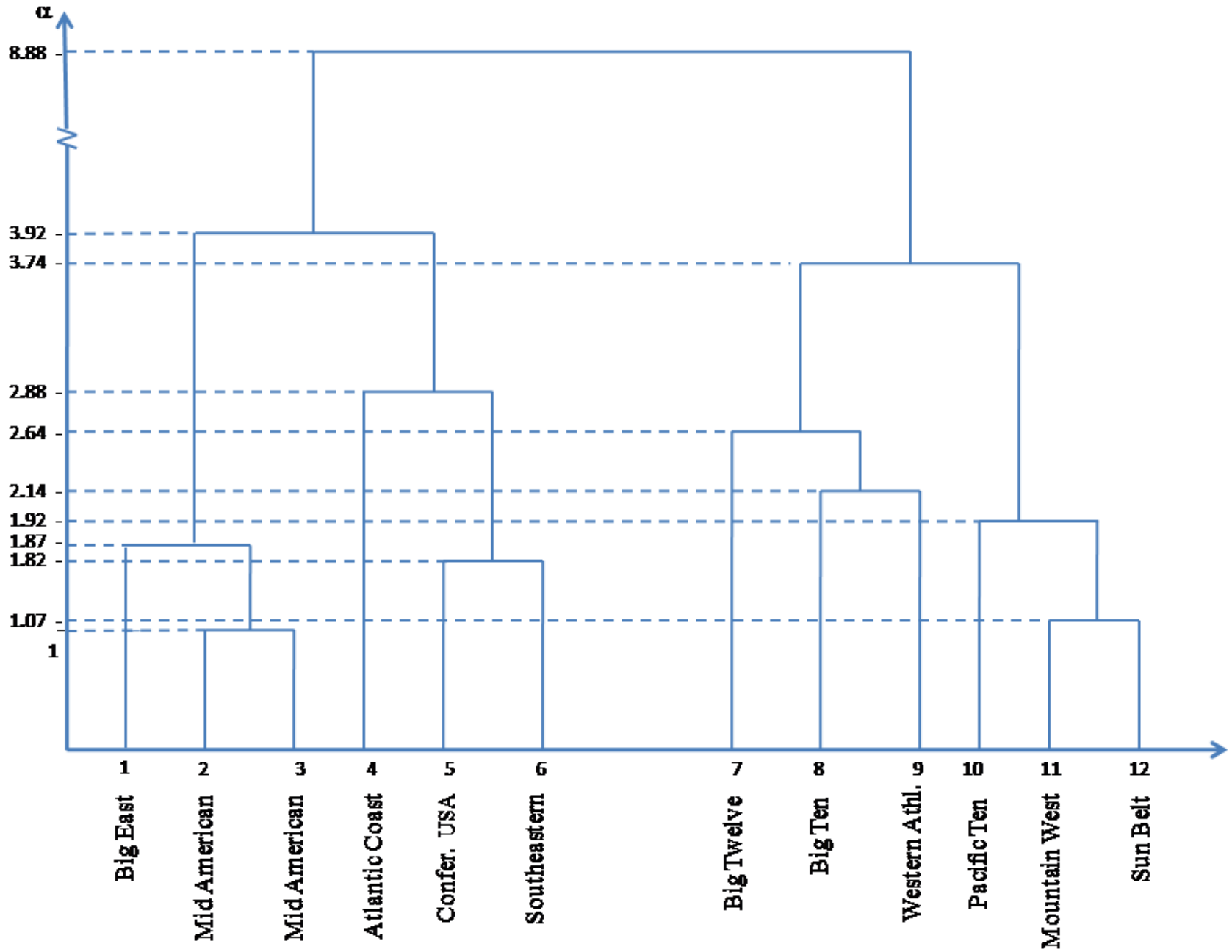
Political books n=114, m=441




α
22
2.95
2.47
2.28
1.73
1.54
1.2
1.04
1
1
2
3
4
5
6
7
8
9
10
communities

Football teams




α
8.88
3.92
3.74
2.88
2.64
2.14
1.92
1.87
1.82
1.07
1
1
2
3
4
5
6
7
8
9
10
11
12
Big East
Mid American
Mid American
Atlantic Coast
Confer. USA
Southeastern
Big Twelve
Big Ten
Western Athl.
Pacific Ten
Mountain West
Sun Belt

Sum-of-squared-distances on networks with VNS

- p-Median problem

$$\begin{array}{ll} \min & \sum_{u,v\in V} d(u,v)^2 x_{uv} \\ \text{s.t.} & \sum_{v\in V} x_{uv} = 1 \quad \forall u \in V \\ & x_{uv} \le y_v \quad \forall u,v \in V \\ & \sum_{v\in V} y_v = p \\ & x_{uv} \in \{0,1\} \quad \forall u,v \in V \\ & y_v \in \{0,1\} \quad \forall v \in V \end{array} \tag{1}$$

- MSSC-Network

$$\begin{array}{ll} \min & \sum_{v\in V} \omega_v d^2(v;\{x_1,\ldots,x_p\}) \\ \text{s.t.} & x_1,\ldots,x_p \in N, \end{array} \tag{1}$$

where, for any $v \in V$, the distance $d(v,P)$ from v to a nonempty finite set P of points the network is defined as the distance to the closest point in P,

$$d(v,\{x_1,\ldots,x_p\}) = \min_{1\le j\le p} d(v,x_j). \tag{1}$$

Instance	n	p	SS	VMSS	%	VNS1	%	VNS2	%	VNS3	%
pmed1	100	5	450233	450233	0.00	450233.00	0.00	450043.94	0.04	450043.94	0.00
pmed2	100	10	271829	256874	5.82	254771.67	0.83	253069.36	0.67	254599.75	0.60
pmed3	100	10	295752	263385	12.29	263040.83	0.13	259643.17	1.29	259643.17	0,00
pmed4	100	20	159678	153963	3.71	151396.38	1.70	151396.38	0.00	149058.42	-1.54
pmed5	100	33	45055	42671	5.59	40281.86	5.93	40281.86	0.00	41699.26	3.52
pmed6	200	5	410360	406195	1.03	406195.00	0.00	386642.24	4.81	397044.84	2.69
pmed7	200	10	222901	221631	0.57	221602.83	0.01	221602.83	0.00	224099.33	1.13
pmed8	200	20	157807	151558	4.12	151170.21	0.26	151170.21	0.00	152541.48	0.91
pmed9	200	40	68886	66525	3.55	63460.34	4.83	63460.34	0.00	73280.84	15.48
pmed10	200	67	16199	15938	1.64	15117.98	5.42	15117.98	0.00	18209.92	20.45
pmed25	500	167	13736	13372	2.72	12536.21	6.67	12536.21	0.00	17018.88	35.76
pmed26	600	5	199503	199503	0.00	199503.00	0.00	199503.00	0.00	199503.00	0.00
pmed27	600	10	147401	147096	0.21	147096.00	0.00	147096.00	0.00	149934.00	1.93
pmed28	600	60	52546	51332	2.36	51175.28	0.31	51175.28	0.00	55310.61	8.08
pmed29	600	120	27143	26017	4.33	25557.24	1.80	25557.24	0.00	30748.51	20.31
pmed30	600	200	12755	12632	0.97	11769.52	7.33	11769.52	0.00	15726.85	33.62
pmed31	700	5	172938	171963	0.57	171963.00	0.00	171963.00	0.00	171963.00	0.00
pmed32	700	10	157283	157283	0.00	157283.00	0.00	157283.00	0.00	161863.00	2.91
pmed33	700	70	49432	47425	4.23	47300.46	0.26	47300.46	0.00	52140.66	10.23
pmed34	700	140	22807	22162	2.91	21665.24	2.29	21665.24	0.00	25854.86	19.34
pmed35	800	5	160564	160564	0,00	160564.00	0.00	160541.91	0.01	163154.00	1.63
pmed36	800	10	153164	153033	0.09	153033.00	0.00	153033.00	0.00	156097.00	2.00
pmed37	800	80	50665	48411	4.66	48195.16	0.45	48195.16	0.00	53333.30	10.66
pmed38	900	5	161102	161102	0.00	161102.00	0.00	161102.00	0.00	162355.00	0.78
pmed39	900	10	126553	125175	1.10	125175.00	0.00	125175.00	0.00	126948.00	1.42
pmed40	900	90	44596	43540	2.43	43278.99	0.60	43278.99	0.00	50427.37	16.52

Table 2: VNS comparison

Test Instance	n	p	Prot in nodes	EMSSC= VMSSC
pmed1	100	5	NO	NO
pmed2	100	10	NO	NO
pmed3	100	10	NO	NO
pmed4	100	20	NO	YES
pmed5	100	33	NO	NO
pmed6	200	5	NO	NO
pmed7	200	10	NO	YES
pmed8	200	20	NO	YES
pmed9	200	40	NO	NO
pmed10	200	67	NO	NO
pmed25	500	167	NO	NO
pmed26	600	5	YES	YES
pmed27	600	10	YES	YES
pmed28	600	60	NO	YES
pmed29	600	120	NO	YES
pmed30	600	200	NO	YES
pmed31	700	5	YES	YES
pmed32	700	10	YES	YES
pmed33	700	70	NO	YES
pmed34	700	140	NO	YES
pmed35	800	5	NO	NO
pmed36	800	10	YES	YES
pmed37	800	80	NO	YES
pmed38	900	5	YES	YES
pmed39	900	10	YES	YES
pmed40	900	90	NO	NO

Table 3: VMSSC vs EMSSC

Beside well known network problems, we wanted to compare results of different clusteri paradigms on classical Ruspini test instance that is usually used for the continuous MSSC.

p	MSSC Network	MSSC continuos	% dev	p-med new	p-med optimal	% dev	MSWP	centers in nodes
2	95250.14	89337.83	6.62	2395.80	2395.80	0.00	2385.55	NO
3	53390.00	51063.48	4.56	1637.81	1619.47	1.13	1609.28	YES
4	13169.00	12881.05	2.24	861.48	861.48	0.00	854.62	YES
5	10572.37	10126.72	4.40	779.68	779.68	0.00	772.39	NO
6	8942.37	8575.41	4.28	715.88	714.65	0.17	706.25	NO
7	7596.37	7126.20	6.60	650.85	650.85	0.00	641.88	NO
8	6537.37	6149.64	6.30	600.01	600.01	0.00	591.06	NO
9	5534.37	5181.65	6.81	555.35	555.35	0.00	546.40	NO
10	4760.32	4446.28	7.06	512.81	512.81	0.00	506.50	NO

Table 4:

Thank you for your attention!

nenadmladenovic12@gmail.com